TD TP 4 PROGRAMMATION ORIENTEE OBJET : HERITAGE

Exercice 1 :Ecosystème

Etape 1

L'objectif de cette étape est de construire une classe *Insecte* permettant de simuler un écosystème très simplifié. Cet écosystème se composera d'insectes qui seront capables de bouger et de se nourrir.

Un objet de type *Insecte* aura comme variables d'instance, une chaine de caractères qui définit le type d'insecte (on se contente de deux types, fourmi et cigale), un entier qui donne le temps écoulé depuis le dernier repas et un entier qui donne le nombre de tours nécessaires pour que l'insecte ait faim.

a. Créer la classe insecte.

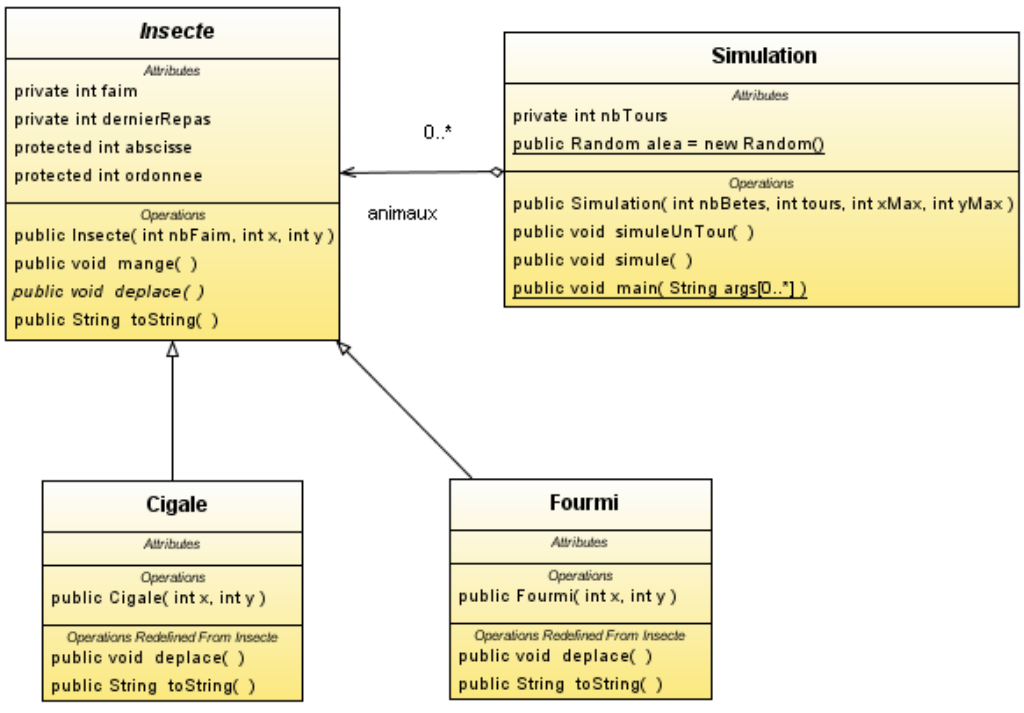
b. Créer les opérateurs d'extraction et de modification pour toutes les variables d'instance (get et set).

c. Ecrire une méthode *mange* qui permet à l'insecte de manger à sa faim.

Etape 2

L'objectif de cette étape est de réaliser une hiérarchie de classes, basée sur la classe insecte écrite précédemment, permettant de simuler un écosystème très simplifié. Cet écosystème sera uniquement composé de cigales et de fourmis qui devront savoir se déplacer et manger. La simulation s'effectue en mode "tour par tour". À chaque tour, tous les insectes effectueront un déplacement et une action. Chaque insecte mangera au bout d'un nombre déterminé de tours.

Il y a plusieurs classes à écrire, l'objectif étant d'obtenir par étapes successives un programme correspondant au diagramme ci-dessous :



- Modifier la classe Insecte (changement des variables d'instance) pour introduire deux nouvelles variables *abscisse* et *ordonnée* qui déterminent la position de l'insecte.
- Écrire la méthode *toString* pour la classe insecte qui retournera une chaîne de caractères correspondant aux caractéristiques de l'insecte.
- Modifier la méthode *mange* pour introduire un test qui vérifie si l'insecte à faim, incrémente le dernier repas et fait manger l'insecte si nécessaire.

Nous allons maintenant ajouter des sous-classes à la classe Insecte pour aboutir au modèle précédent.

- Ecrire deux sous-classes de la classe Insecte : Cigale et Fourmi. Chacune de ces classes possédera un constructeur qui appellera celui de Insecte, sachant qu'une cigale a faim tous les 5 tours, et une fourmi tous les 7 tours.
- Écrire la méthode toString des deux sous-classes. Cette méthode retournera le message "Je suis une cigale, " ou "Je suis une fourmi, " suivi du message de la classe Insecte.

Nous allons écrire la classe *Simulation*, qui simule un certain nombre de tours pour un tableau d'insectes. Le but étant de tester les méthodes des autres classes. Les variables d'instances seront un entier *nbrInsectes* représentant le nombre d'insectes dans la simulation, un entier *nbrTour* représentant le nombre te tour à effectuer et deux entiers *xMax* et *yMax* représentant les dimensions de notre domaine de simulation.

- Ecrire le constructeur de cette classe.
- Ecrire une méthode simule qui affiche l'insecte le déplace et le fait manger si cela est nécessaire.
- Tester la classe *Simulation* dans le main.

Exercice 2 :Gestion de véhicules :

Le but est d'écrire une hiérarchie de classes basée sur une classe *Vehicule*.

- Ecrire une classe *Vehicule* qui a comme représentation interne trois entiers (nombre de roues, nombre de places et kilométrage).
- Ecrire une méthode *toString* qui retourne une chaine de caractères qui donne la description d'un objet *Vehicule*.
- Tester la classe *Vehicule* dans le main.

- Ecrire deux sous-classes *Voiture* et *Moto*. Les deux sous classes auront comme variables internes un entier *Poids*, une chaine de caractères *Couleur* et un entier *Puissance*.
- Ecrire pour chaque sous-classe une méthode *toString* qui, en plus de la description donnée par la méthode *toString* de la classe *Vehicule*, donne la description de la sous-classe.
- Tester les classes *Voiture* et *Moto* dans le main.

- Ecrire une sous-classe de la classe *Voiture* (lui donner comme nom une marque de voiture). Pour la représentation interne de cette sous-classe se présente sous forme de deux réels *Consommation* et *Capacite_Reservoir*.
- Ecrire les méthodes *get* et *set* pour toutes les variables d'instance.
- Tester cette classe dans le main.